林轩田《机器学习基石》课程笔记9 -- Linear Regression

作者：红色石头　　公众号：AI有道（id：redstonewill）

上节课，我们主要介绍了在有noise的情况下，VC Bound理论仍然是成立的。同时，介绍了不同的error measure方法。本节课介绍机器学习最常见的一种算法：Linear Regression.

一、线性回归问题

在之前的Linear Classification课程中，讲了信用卡发放的例子，利用机器学习来决定是否给用户发放信用卡。本节课仍然引入信用卡的例子，来解决给用户发放信用卡额度的问题，这就是一个线性回归（Linear Regression）问题。

Linear Regression Hypothesis

age	23 years
annual salary	NTD 1,000,000
year in job	0.5 year
current debt	200,000

- For $\mathbf{x} = (x_0, x_1, x_2, \cdots, x_d)$ 'features of customer', approximate the desired credit limit with a weighted sum:

$$y \approx \sum_{i=0}^{d} w_i x_i$$

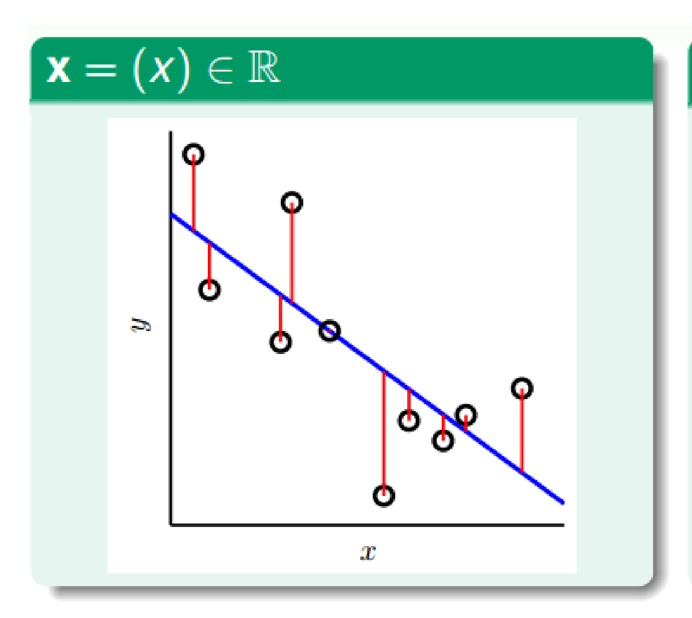
- linear regression hypothesis: $h(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$

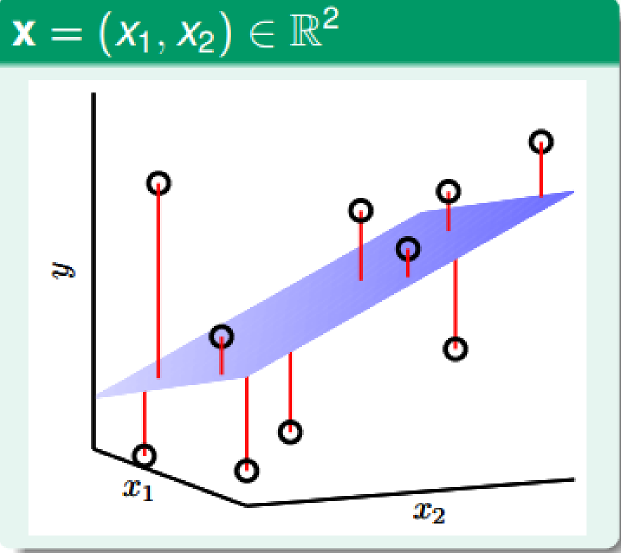
$h(\mathbf{x})$: like perceptron, but without the sign

令用户特征集为d维的X，加上常数项，维度为$d+1$，与权重w的线性组合即为Hypothesis,记为$h(x)$。线性回归的预测函数取值在整个实数空间，这跟线性分类不

同。

$$h(x) = w^T X$$





linear regression:
find lines/hyperplanes with small residuals

根据上图，在一维或者多维空间里，线性回归的目标是找到一条直线（对应一维）、一个平面（对应二维）或者更高维的超平面，使样本集中的点更接近它，也就是残留误差Residuals最小化。

一般最常用的错误测量方式是基于最小二乘法，其目标是计算误差的最小平方和对应的权重w，即上节课介绍的squared error：

popular/historical error measure:

squared error $\text{err}(\hat{y}, y) = (\hat{y} - y)^2$

in-sample

$$E_{\text{in}}(h\mathbf{w}) = \frac{1}{N}\sum_{n=1}^{N}(\underbrace{h(\mathbf{x}_n)}_{\mathbf{w}^T\mathbf{x}_n} - y_n)^2$$

out-of-sample

$$E_{\text{out}}(\mathbf{w}) = \underset{(\mathbf{x},y)\sim P}{\mathcal{E}}(\mathbf{w}^T\mathbf{x} - y)^2$$

这里提一点，最小二乘法可以解决线性问题和非线性问题。线性最小二乘法的解是closed-form，即$X = (A^T A)^{-1} A^T y$，而非线性最小二乘法没有closed-form，通常

用迭代法求解。本节课的解就是closed-form的。关于最小二乘法的一些介绍，请参见我的另一篇博文：

最小二乘法和梯度下降法的一些总结

二、线性回归算法

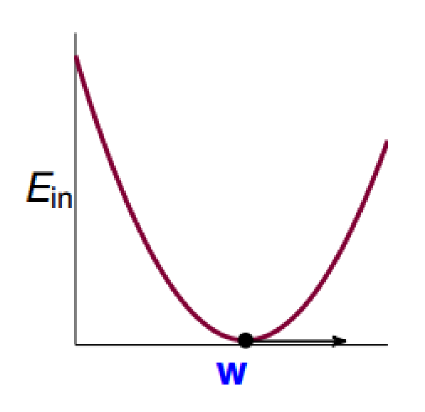
样本数据误差E_{in}是权重w的函数，因为X和y都是已知的。我们的目标就是找出合适的w，使E_{in}能够最小。那么如何计算呢？

首先，运用矩阵转换的思想，将E_{in}计算转换为矩阵的形式。

$$
\begin{aligned}
E_{in}(\mathbf{w}) &= \frac{1}{N}\sum_{n=1}^{N}(\mathbf{w}^T\mathbf{x}_n - y_n)^2 = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n^T\mathbf{w} - y_n)^2 \\
&= \frac{1}{N}\left\| \begin{matrix} \mathbf{x}_1^T\mathbf{w} - y_1 \\ \mathbf{x}_2^T\mathbf{w} - y_2 \\ \cdots \\ \mathbf{x}_N^T\mathbf{w} - y_N \end{matrix} \right\|^2 \\
&= \frac{1}{N}\left\| \begin{bmatrix} --\mathbf{x}_1^T-- \\ --\mathbf{x}_2^T-- \\ \cdots \\ --\mathbf{x}_N^T-- \end{bmatrix} \mathbf{w} - \begin{bmatrix} y_1 \\ y_2 \\ \cdots \\ y_N \end{bmatrix} \right\|^2 \\
&= \frac{1}{N}\| \underbrace{\mathrm{X}}_{N\times d+1} \underbrace{\mathbf{w}}_{d+1\times 1} - \underbrace{\mathbf{y}}_{N\times 1} \|^2
\end{aligned}
$$

然后，对于此类线性回归问题，$E_{in}(w)$一般是个凸函数。凸函数的话，我们只要找到一阶导数等于零的位置，就找到了最优解。那么，我们将E_w对每个$w_i, i = 0, 1, \cdots, d$求偏导，偏导为零的w_i，即为最优化的权重值分布。

$$\min_{\mathbf{w}} E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\|\mathrm{X}\mathbf{w} - \mathbf{y}\|^2$$



- $E_{\text{in}}(\mathbf{w})$: continuous, differentiable, **convex**
- necessary condition of 'best' $\mathbf{w}$

$$\nabla E_{\text{in}}(\mathbf{w}) \equiv \begin{bmatrix} \frac{\partial E_{\text{in}}}{\partial w_0}(\mathbf{w}) \\ \frac{\partial E_{\text{in}}}{\partial w_1}(\mathbf{w}) \\ \cdots \\ \frac{\partial E_{\text{in}}}{\partial w_d}(\mathbf{w}) \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \cdots \\ 0 \end{bmatrix}$$

—not possible to 'roll down'

task: find $\mathbf{w}_{\text{LIN}}$ such that $\nabla E_{\text{in}}(\mathbf{w}_{\text{LIN}}) = \mathbf{0}$

根据梯度的思想，对E_w进行矩阵话求偏导处理：

The Gradient $\nabla E_{\text{in}}(\mathbf{w})$

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\|\mathrm{X}\mathbf{w} - \mathbf{y}\|^2 = \frac{1}{N}\left(\mathbf{w}^T \underbrace{\mathrm{X}^T\mathrm{X}}_{\mathrm{A}} \mathbf{w} - 2\mathbf{w}^T \underbrace{\mathrm{X}^T\mathbf{y}}_{\mathbf{b}} + \underbrace{\mathbf{y}^T\mathbf{y}}_{c}\right)$$

one w only

$$E_{\text{in}}(w) = \frac{1}{N}\left(aw^2 - 2bw + c\right)$$

$$\nabla E_{\text{in}}(w) = \frac{1}{N}(2aw - 2b)$$

simple! :-)

vector $\mathbf{w}$

$$E_{\text{in}}(\mathbf{w}) = \frac{1}{N}\left(\mathbf{w}^T\mathrm{A}\mathbf{w} - 2\mathbf{w}^T\mathbf{b} + c\right)$$

$$\nabla E_{\text{in}}(\mathbf{w}) = \frac{1}{N}(2\mathrm{A}\mathbf{w} - 2\mathbf{b})$$

similar (derived by definition)

$$\nabla E_{\text{in}}(\mathbf{w}) = \frac{2}{N}\left(\mathrm{X}^T\mathrm{X}\mathbf{w} - \mathrm{X}^T\mathbf{y}\right)$$

令偏导为零，最终可以计算出权重向量w为：

Optimal Linear Regression Weights

task: find $\mathbf{w}_{\text{LIN}}$ such that $\frac{2}{N}\left(X^T X \mathbf{w} - X^T \mathbf{y}\right) = \nabla E_{\text{in}}(\mathbf{w}) = \mathbf{0}$

invertible $X^T X$

- **easy!** unique solution

$$\mathbf{w}_{\text{LIN}} = \underbrace{\left(X^T X\right)^{-1} X^T}_{\text{pseudo-inverse } X^\dagger} \mathbf{y}$$

- often the case because $N \gg d+1$

singular $X^T X$

- **many** optimal solutions
- one of the solutions

$$\mathbf{w}_{\text{LIN}} = X^\dagger \mathbf{y}$$

by defining $X^\dagger$ in other ways

practical suggestion:
use **well-implemented $\dagger$ routine**
instead of $\left(X^T X\right)^{-1} X^T$
for numerical stability when **almost-singular**

最终，我们推导得到了权重向量$w = (X^T X)^{-1} X^T y$，这是上文提到的closed-form解。其中，$(X^T X)^{-1} X^T$又称为伪逆矩阵pseudo-inverse，记为X^+，维度是(d+1)xN。

但是，我们注意到，伪逆矩阵中有逆矩阵的计算，逆矩阵$(X^T X)^{-1}$是否一定存在？一般情况下，只要满足样本数量N远大于样本特征维度d+1，就能保证矩阵的逆是存在的，称之为非奇异矩阵。但是如果是奇异矩阵，不可逆怎么办呢？其实，大部分的计算逆矩阵的软件程序，都可以处理这个问题，也会计算出一个逆矩阵。所以，一般伪逆矩阵是可解的。

三、泛化问题

现在，可能有这样一个疑问，就是这种求解权重向量的方法是机器学习吗？或者说这种方法满足我们之前推导VC Bound，即是否泛化能力强$E_{in} \approx E_{out}$？

Is Linear Regression a 'Learning Algorithm'?

$$\mathbf{w}_{\text{LIN}} = \mathrm{X}^{\dagger}\mathbf{y}$$

No!

- analytic (**closed-form**) solution, 'instantaneous'
- not improving E_{in} nor E_{out} iteratively

Yes!

- good E_{in}? **yes, optimal!**
- good E_{out}? **yes, finite d_{VC} like perceptrons**
- improving iteratively? **somewhat, within an iterative pseudo-inverse routine**

if $E_{\text{out}}(\mathbf{w}_{\text{LIN}})$ is good, **learning 'happened'!**

有两种观点：1、这不属于机器学习范畴。因为这种closed-form解的形式跟一般的机器学习算法不一样，而且在计算最小化误差的过程中没有用到迭代。2、这属于机器学习范畴。因为从结果上看，E_{in}和E_{out}都实现了最小化，而且实际上在计算逆矩阵的过程中，也用到了迭代。

其实，只从结果来看，这种方法的确实现了机器学习的目的。下面通过介绍一种更简单的方法，证明linear regression问题是可以通过线下最小二乘法方法计算得到好的E_{in}和E_{out}的。

Benefit of Analytic Solution: 'Simpler-than-VC' Guarantee

$$\overline{E_{in}} = \underset{\mathcal{D}\sim P^N}{\mathcal{E}}\left\{E_{in}(\mathbf{w}_{LIN} \text{ w.r.t. } \mathcal{D})\right\} \overset{\text{to be shown}}{=} \text{noise level} \cdot \left(1 - \frac{d+1}{N}\right)$$

$$E_{in}(\mathbf{w}_{LIN}) = \frac{1}{N}\|\mathbf{y} - \underbrace{\hat{\mathbf{y}}}_{\text{predictions}}\|^2 = \frac{1}{N}\|\mathbf{y} - \mathrm{X}\underbrace{\mathrm{X}^{\dagger}\mathbf{y}}_{\mathbf{w}_{LIN}}\|^2$$

$$= \frac{1}{N}\|(\underbrace{\mathrm{I}}_{\text{identity}} - \mathrm{X}\mathrm{X}^{\dagger})\mathbf{y}\|^2$$

call $\mathrm{X}\mathrm{X}^{\dagger}$ the hat matrix H
because it puts ∧ on **y**

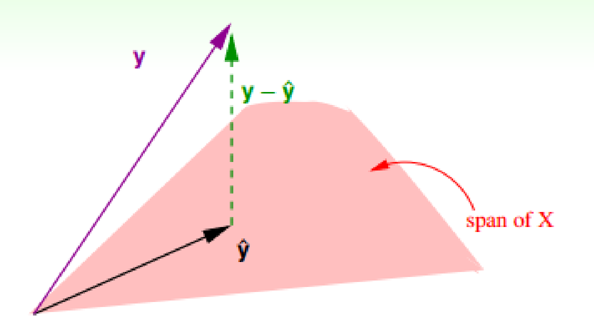
首先，我们根据平均误差的思想，把$E_{in}(w_{LIN})$写成如图的形式，经过变换得到:

$$E_{in}(w_{LIN}) = \frac{1}{N}||(I - XX^{+})y||^2 = \frac{1}{N}||(I - H)y||^2$$

我们称XX^{+}为帽子矩阵，用H表示。

下面从几何图形的角度来介绍帽子矩阵H的物理意义。

Geometric View of Hat Matrix



in $\mathbb{R}^N$

- $\hat{\mathbf{y}} = \mathrm{X}\mathbf{w}_{\mathrm{LIN}}$ within the span of X columns
- $\mathbf{y} - \hat{\mathbf{y}}$ smallest: $\mathbf{y} - \hat{\mathbf{y}} \perp$ span
- H: project $\mathbf{y}$ to $\hat{\mathbf{y}} \in$ span
- $\mathrm{I} - \mathrm{H}$: transform $\mathbf{y}$ to $\mathbf{y} - \hat{\mathbf{y}} \perp$ span

claim: $\mathrm{trace}(\mathrm{I} - \mathrm{H}) = N - (d + 1)$. **Why? :-)**

图中，y是N维空间的一个向量，粉色区域表示输入矩阵X乘以不同权值向量w所构成的空间，根据所有w的取值，预测输出都被限定在粉色的空间中。向量$\hat{y}$就是粉色空间中的一个向量，代表预测的一种。y是实际样本数据输出值。

机器学习的目的是在粉色空间中找到一个$\hat{y}$，使它最接近真实的y，那么我们只要将y在粉色空间上作垂直投影即可，投影得到的$\hat{y}$即为在粉色空间内最接近y的向量。这样即使平均误差$\overline{E}$最小。

从图中可以看出，$\hat{y}$是y的投影，已知$\hat{y} = Hy$，那么H表示的就是将y投影到$\hat{y}$的一种操作。图中绿色的箭头$y - \hat{y}$是向量y与$\hat{y}$相减，$y - \hat{y}$垂直于粉色区域。已知$(I - H)y = y - \hat{y}$那么I-H表示的就是将y投影到$y - \hat{y}$即垂直于粉色区域的一种操作。这样的话，我们就赋予了H和I-H不同但又有联系的物理意义。

这里trace(I-H)称为I-H的迹，值为N-(d+1)。这条性质很重要，一个矩阵的 trace等于该矩阵的所有特征值(Eigenvalues)之和。下面给出简单证明：

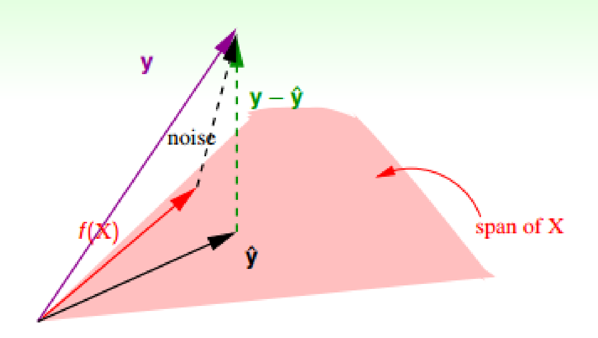
$$trace(I - H) = trace(I) - trace(H)$$
$$= N - trace(XX^{+}) = N - trace(X(X^TX)^{-1}X^T$$
$$= N - trace(X^TX(X^TX)^{-1}) = N - trace(I_{d+1}$$
$$= N - (d + 1)$$

介绍下该I-H这种转换的物理意义：原来有一个有N个自由度的向量y，投影到一个有d+1维的空间x（代表一列的自由度，即单一输入样本的参数，如图中粉色区域），而

余数剩余的自由度最大只有N-(d+1)种。

在存在noise的情况下，上图变为：

An Illustrative ‘Proof’



- if **y** comes from some ideal $f(X) \in$ span plus **noise**
- **noise** transformed by $I - H$ to be $\mathbf{y} - \hat{\mathbf{y}}$

$$E_{\text{in}}(\mathbf{w}_{\text{LIN}}) = \frac{1}{N}\|\mathbf{y} - \hat{\mathbf{y}}\|^2 = \frac{1}{N}\|(I - H)\mathbf{noise}\|^2 = \frac{1}{N}(N - (d+1))\|\mathbf{noise}\|^2$$

$$\overline{E_{\text{in}}} = \mathbf{noise}\ \text{level} \cdot \left(1 - \frac{d+1}{N}\right)$$

$$\overline{E_{\text{out}}} = \mathbf{noise}\ \text{level} \cdot \left(1 + \frac{d+1}{N}\right) \text{ (complicated!)}$$

图中，粉色空间的红色箭头是目标函数f(x)，虚线箭头是noise，可见，真实样本输出y由f(x)和noise相加得到。由上面推导，已知向量y经过I-H转换为$y-\hat{y}$，而noise与y是线性变换关系，那么根据线性函数知识，我们推导出noise经过I-H也能转换为$y-\hat{y}$。则对于样本平均误差，有下列推导成立：

$$E_{in}(w_{LIN}) = \frac{1}{N}||y-\hat{y}||^2 = \frac{1}{N}||(I-H)noise||^2 = \frac{1}{N}(N-(d+1))||noise||^2$$

即

$$\overline{E}_{in} = noiselevel * (1 - \frac{d+1}{N})$$

同样，对E_{out}有如下结论：

$$\overline{E}_{out} = noiselevel * (1 + \frac{d+1}{N})$$

这个证明有点复杂，但是我们可以这样理解：$\overline{E}_{in}$与$\overline{E}_{out}$形式上只差了$\frac{(d+1)}{N}$项，从哲

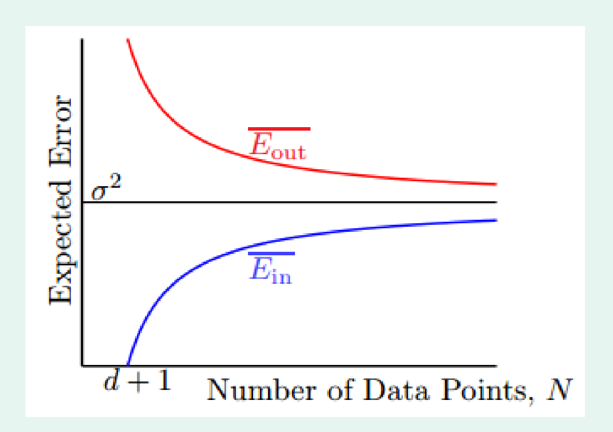
学上来说，$\overline{E}_{in}$是我们看得到的样本的平均误差，如果有noise，我们把预测往noise那边偏一点，让$\overline{E}_{in}$好看一点点，所以减去$\frac{(d+1)}{N}$项。那么同时，新的样本$\overline{E}_{out}$是我们看不到的，如果noise在反方向，那么$\overline{E}_{out}$就应该加上$\frac{(d+1)}{N}$项。

我们把$\overline{E}_{in}$与$\overline{E}_{out}$画出来，得到学习曲线：

The Learning Curve

$\overline{E_{\text{out}}} = \textbf{noise} \text{ level} \cdot \left(1 + \frac{d+1}{N}\right)$

$\overline{E_{\text{in}}} = \textbf{noise} \text{ level} \cdot \left(1 - \frac{d+1}{N}\right)$



- both converge to σ^2 (**noise** level) for $N \to \infty$
- expected generalization error: $\frac{2(d+1)}{N}$
  —**similar to worst-case guarantee from VC**

linear regression (LinReg):
learning 'happened'!

当N足够大时，$\overline{E}_{in}$与$\overline{E}_{out}$逐渐接近，满足$\overline{E}_{in} \approx \overline{E}_{out}$，且数值保持在noise level。这就类似VC理论，证明了当N足够大的时候，这种线性最小二乘法是可以进行机器学习的，算法有效！

四、Linear Regression方法解决Linear Classification问题

之前介绍的Linear Classification问题使用的Error Measure方法用的是0/1 error，那么Linear Regression的squared error是否能够应用到Linear Classification问题？

Linear Classification vs. Linear Regression

Linear Classification

$$\mathcal{Y} = \{-1, +1\}$$
$$h(\mathbf{x}) = \text{sign}(\mathbf{w}^T\mathbf{x})$$
$$\text{err}(\hat{y}, y) = [\![\hat{y} \neq y]\!]$$

NP-hard to solve in general

Linear Regression

$$\mathcal{Y} = \mathbb{R}$$
$$h(\mathbf{x}) = \mathbf{w}^T\mathbf{x}$$
$$\text{err}(\hat{y}, y) = (\hat{y} - y)^2$$

efficient analytic solution

$\{-1, +1\} \subset \mathbb{R}$: linear regression for classification?

1. run LinReg on binary classification data $\mathcal{D}$ (**efficient**)
2. return $g(\mathbf{x}) = \text{sign}(\mathbf{w}_{\text{LIN}}^T\mathbf{x})$

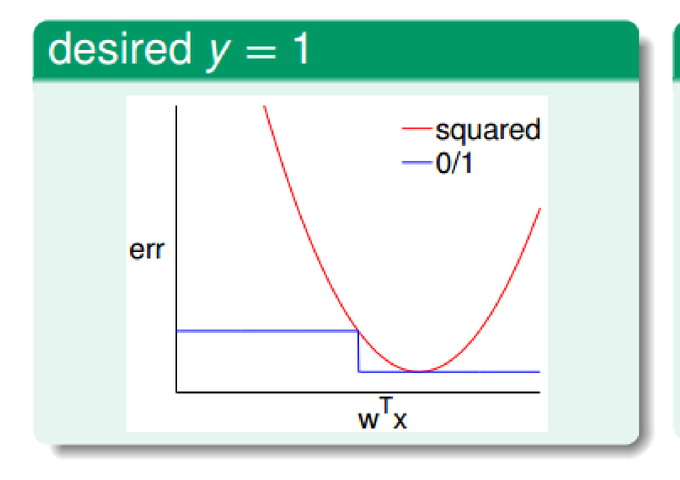
but explanation of this **heuristic**?

下图展示了两种错误的关系，一般情况下，squared error曲线在0/1 error曲线之上。即 $err_{0/1} \leq err_{sqr}$.

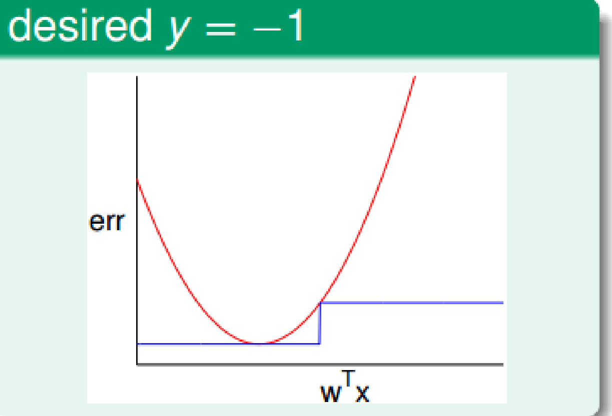
Relation of Two Errors

$$\text{err}_{0/1} = [\![\text{sign}(\mathbf{w}^T\mathbf{x}) \neq y]\!] \qquad \text{err}_{\text{sqr}} = \left(\mathbf{w}^T\mathbf{x} - y\right)^2$$

desired $y = 1$



desired $y = -1$



$$\text{err}_{0/1} \leq \text{err}_{\text{sqr}}$$

根据之前的VC理论，E_{out}的上界满足：

Linear Regression for Binary Classification

$\text{err}_{0/1} \leq \text{err}_{\text{sqr}}$

$$\text{classification } E_{\text{out}}(\mathbf{w}) \overset{VC}{\leq} \text{classification } E_{\text{in}}(\mathbf{w}) + \sqrt{\cdots\cdots}$$
$$\leq \text{regression } E_{\text{in}}(\mathbf{w}) + \sqrt{\cdots\cdots}$$

- (loose) upper bound err_{sqr} as $\widehat{\text{err}}$ to approximate $\text{err}_{0/1}$
- trade **bound tightness** for **efficiency**

$\mathbf{w}_{\text{LIN}}$: useful baseline classifier,
or as **initial PLA/pocket vector**

从图中可以看出，用err_{sqr}代替$err_{0/1}$，E_{out}仍然有上界，只不过是上界变得宽松了。也就是说用线性回归方法仍然可以解决线性分类问题，效果不会太差。二元分类问题得到了一个更宽松的上界，但是也是一种更有效率的求解方式。

五、总结

本节课，我们主要介绍了Linear Regression。首先，我们从问题出发，想要找到一条直线拟合实际数据值；然后，我们利用最小二乘法，用解析形式推导了权重w的closed-form解；接着，用图形的形式得到$E_{out}-E_{in}\approx\frac{2(N+1)}{N}$，证明了linear regression是可以进行机器学习的，；最后，我们证明linear regressin这种方法可以用在binary classification上，虽然上界变宽松了，但是仍然能得到不错的学习方法。

注明：

文章中所有的图片均来自台湾大学林轩田《机器学习基石》课程